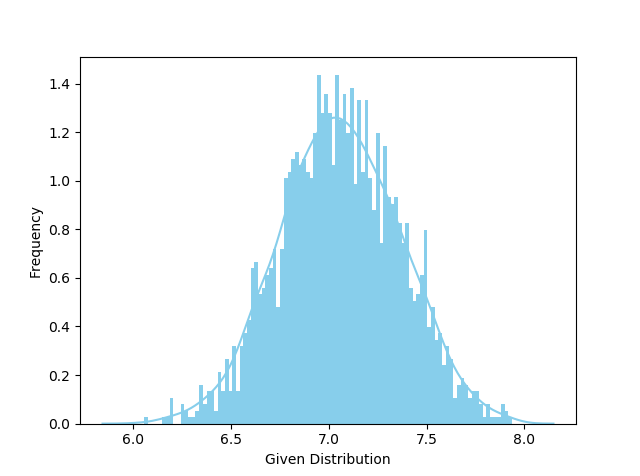
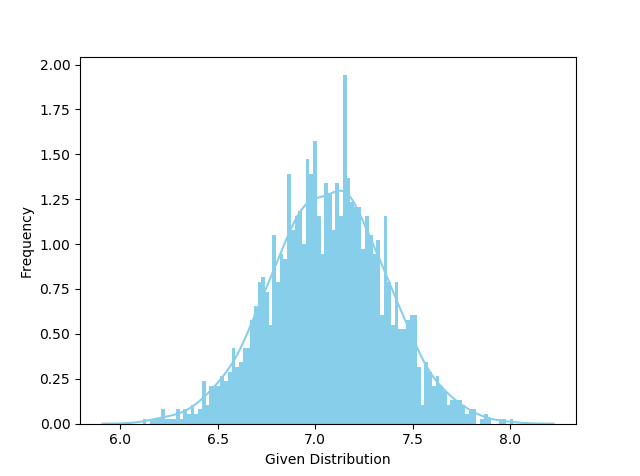
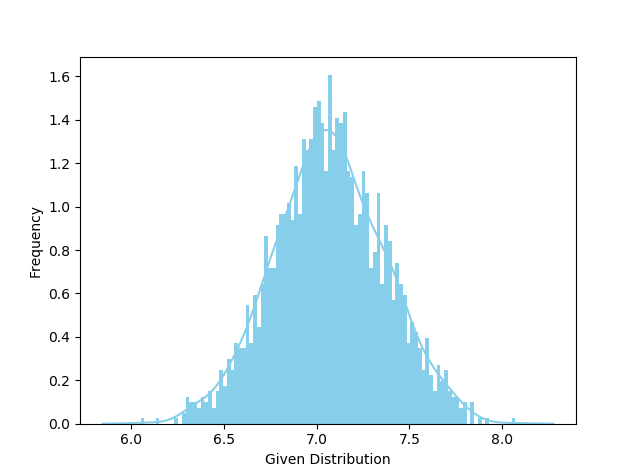
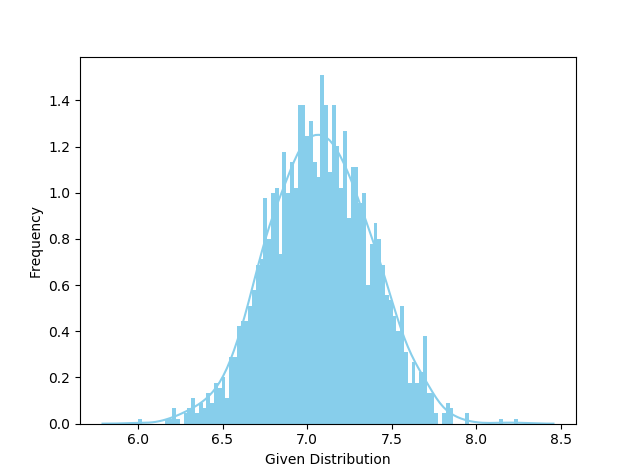
## 2 Maximum Likelihood Estimation of Model Parameters

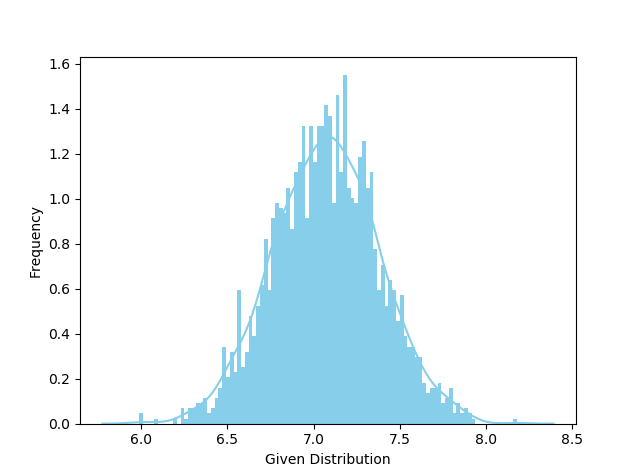
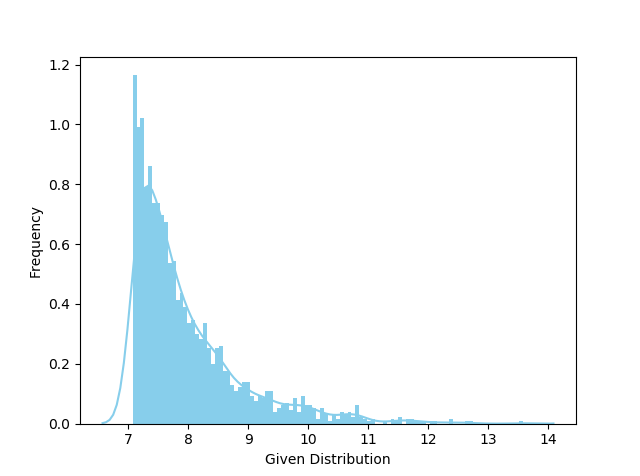
### 2.1 First scenario

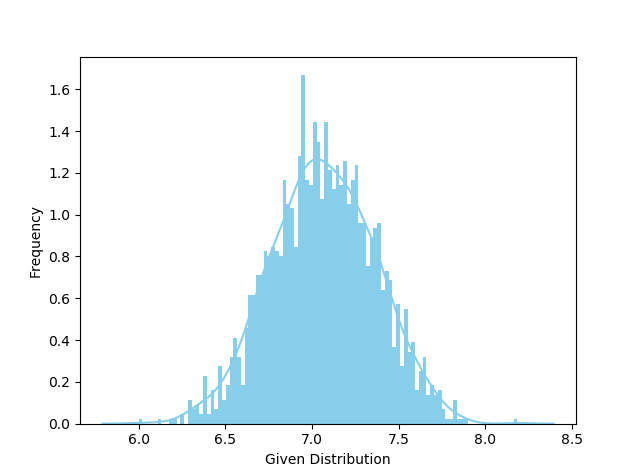
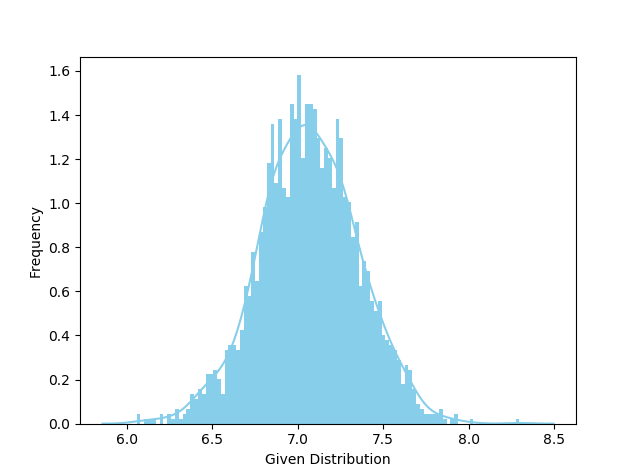
For the first scenario we plotted the frequency of the given distribution of the points in the reference\_measurement\_1 so that we could analyze its distribution. The graphs for the various anchors obtained are the following:



As we can see the distribution is a typical Gaussian distribution.

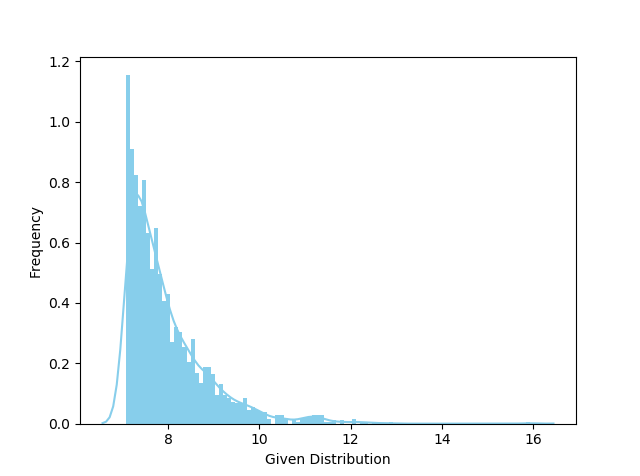
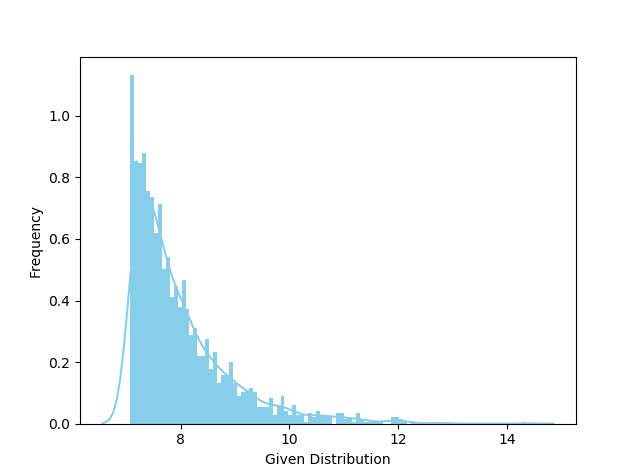
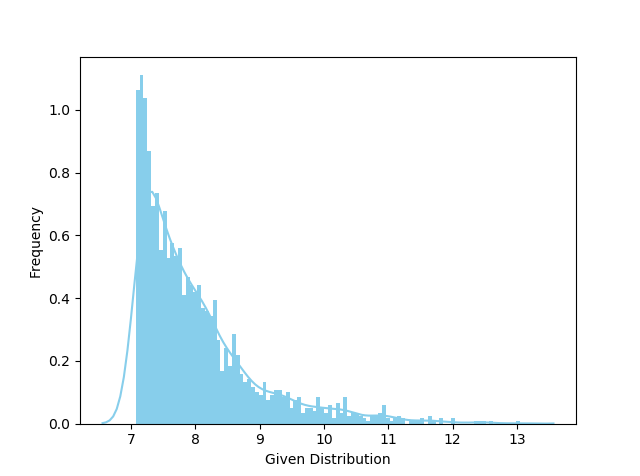
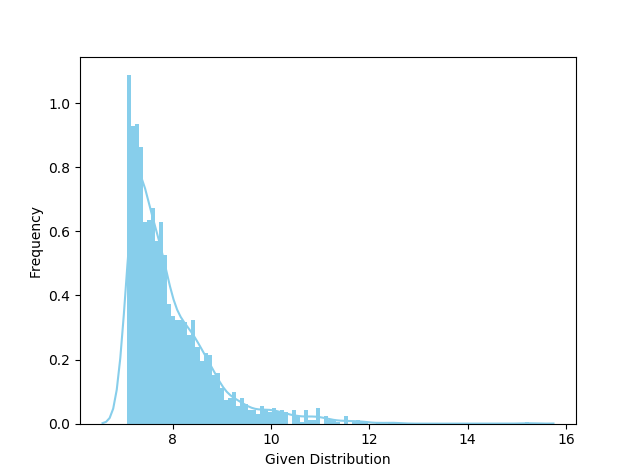
### 2.2 Second Scenario

We plotted the frequency of the distribution as in the first scenario with the following output: 



By looking at the plots it’s pretty clear that the measurements of the first anchor are following the Exponential model and the other 3 the Gaussian as in the first scenario.

### 2.3 Third Scenario

As in the others scenarios we first looked at the frequency of the distribution in the plots:

As for the measurements for the first anchor in scenario 2 the distribution of all the measurements in scenario three follow the Exponential model.

### 2.4 Estimation of parameter

Now that we know the model of each anchor in each scenario we can calculate the parameter θ (λi or σi depending on the model) for each anchor. For the Guassian model we calculate the parameter using the numpy command np.cov wich take the measuraments for the anchor and return the covariance.

For the Exponential model we calculated the parameter using the Maximum Likelihood Estimation method which steps are shown below.

We start with the definition of exponentially distributed.

Its likelihood function is

The derivative with respect to must equal zero

To calculate the rate we solve for

Finally we get

### 2.5 Parameters

#### Scenario 1

#### Scenario 2

#### Scenario 3

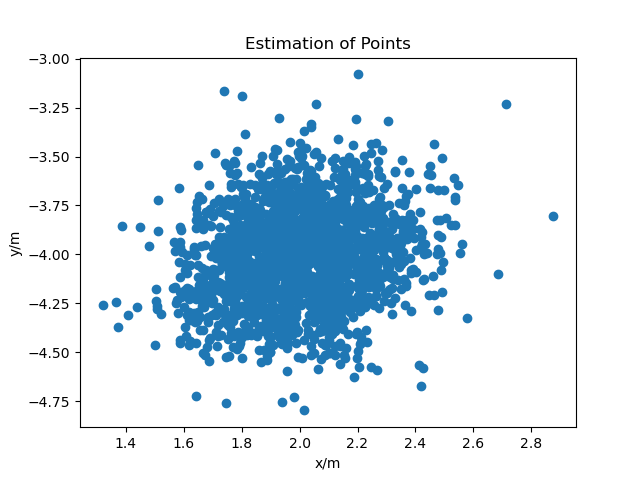
## 3 Estimation of the Position

### 3.1 Least-Squares Estimation of the Position

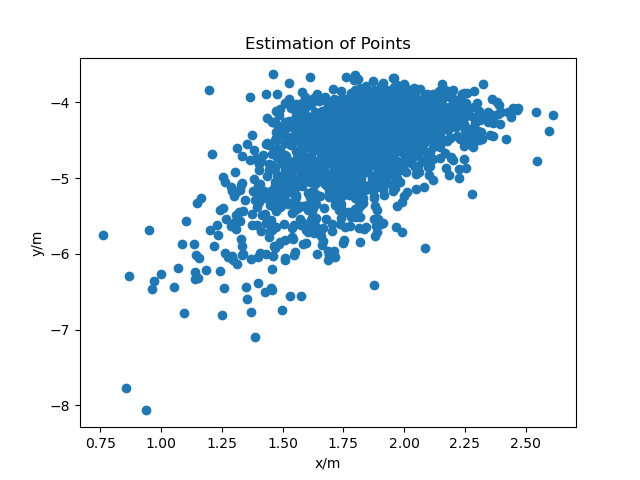
// TODO

### 3.2 Gauss-Newton Algorithm for Position Estimation

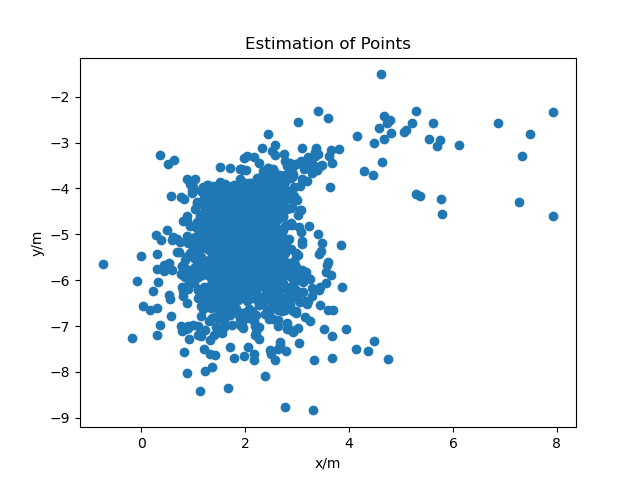
First of all we have calculated the derivate for the Jacobian Matrix. dn (ai , p) is a costant and so in the derivative is canceled. Because of that the result is only the distance between the coordinates of the measured point and the guessed p\_start which is randomly calculated based on the uniform distribution within the square spanned by the anchor points. Now that we have the Jacobian Matrix and the formula given in the assignment sheet we can calculate p(t + 1) . We calculate p(t + 1) until tolerance or the maximum number of interaction is reached and then return the founded point to the calling function. We’ve chosen a tolerance of 10^(-4) for having a good precision but not too many steps. With this tolerance our function never reach more than 9 interaction, we’ve chosen 8 as max\_iter for maintaining a good precision but letting the function exit because of reaching the maximum number of interaction only a few times. After we’ve estimated one point for each measurement and saved all the points into an array we’ve calculated the squared errors between the founded point and p\_true. We saved all the errors into a list and with the statistics functions stastistics.mean() and statistics.variance() we’ve calculated the mean and variance of the errors for each scenario. After the calculations have been completed we called the function ecdf and printed out the graph of the CDF-function and we printed out different type of graph for analyzing our results. Our plots and our analysis is in the pages below. Finally we performed all these steps again excluding the first anchor from the calculus for the second scenario.

These are the resulting plots of all the estimated points for each scenario:

**Scenario 1:** As we can see the distribution of the results of the least squares estimation for the first scenario looks pretty homogeneous distributed. This is probably due to the Gaussian model adopted



**Scenario 2:** As in the first scenario the points are pretty homogeneous distributed except for some points at the bottom left of the graph. This is probably due the errors in measurements of the first anchor ( the one who follow the Exponential model).



**Scenario 3:** The points here are grouped in a position the isn’t the position of p\_true (2, -4) and there are points that are pretty far away from the others. This is probably due the Exponential model used in this scenario

Here we have another perspective of the resulting points of the least squares estimation of position compared to p\_true and the anchors (Please note that not all the resulting points are shown in the graphs for having a better perspective, the distribution of all points is shown in the graphs above):

